Reverse Engineering to Achieve Maintainable WWW Sites

Cornelia Boldyreff[1] **and Richard Kewish**
R.I.S.E.
Department of Computer Science
University of Durham
Durham, DH1 3LE, U.K.
+44 191 374 2638
cornelia.boldyreff@durham.ac.uk

ABSTRACT
The growth of the World Wide Web and the accelerated development of web sites and associated web technologies has resulted in a variety of maintenance problems. The maintenance problems associated with web sites and the WWW are examined.
It is argued that currently web sites and the WWW lack both data abstractions and structures that could facilitate maintenance. A system to analyse existing web sites and extract duplicated content and style is described here. In designing the system, existing Reverse Engineering techniques have been applied, and a case for further application of these techniques is made in order to prepare sites for their inevitable evolution in future.

Keywords
Web site maintenance, web analysis, detection of duplicated web content, re-structuring, data abstraction

1 BACKGROUND

Since the World Wide Web (WWW, or web) first became widely accepted, it has grown to be a huge information and storage medium. The speed and breadth of its growth and its inherent complexity have meant that the web has not seen many of the rigorous development and maintenance principles applied to its contents and applications that traditional Software Engineering applies to more conventional large scale software applications [1,2].

In the 1970s and 80s, computer scientists started to consider software as another engineered product and began to apply the same stringent design, production, and maintenance procedures that more conventional engineering disciplines already employed. The web is now being considered in the same way [2] and concepts developed in traditional Software Engineering, for example, Lehman's Laws [3] are beginning to be studied with respect to the development of web sites over time [1]. However, this does not address the issue of web site maintenance which has not been as widely considered as the development of a web site [4,5], but is equally important now that the web has grown so large and has become an integral part of modern society.

The current web site format is a collection of files containing several different types of information that can be retrieved and viewed using a web browser. Typically documents are constructed using the Hypertext Markup Language (HTML) which defines the style, structure, and content of the web pages. These pages can also contain embedded elements and links to other files of the same or different formats such as images or Java applets.

On a conventional web site, each page is stored as a separate file in a hierarchical directory structure. These are stored on one or more servers that run software to implement the Hypertext Transfer Protocol (HTTP) which processes client requests (e.g. from distributed web browsers) and returns the relevant files. However, the haphazard manner in which the web has come into being has had a profound influence on the its current state and as a result much of the existing web is very hard to maintain and has not been subjected to systemic or routine maintenance.

The current system of storage for web sites causes several problems for maintaining the site; these are as follows:

- the contents of individual files may be duplicated across several pages, possibly causing inconsistent data throughout the site;
- the page styles may not be consistent across the site giving an unprofessional look to the site and possibly making it difficult to navigate;

[1] Corresponding author

Proceedings of the Eighth Working Conference On Reverse Engineering (WCRE'01)
0-7695-1303-4/02 $17.00 © 2002 IEEE

- links and references on web sites are notoriously volatile so they require frequent checking and updating [6], a task that is impeded by decentralisation of the web site contents and distribution of its maintenance.

It is now beginning to be accepted that a web site can be regarded in much the same way as a large-scale conventional software system [2,7], with many of the same maintenance requirements as well as some of the specification and design requirements. Therefore, in a similar manner to the way that software models and programming language structures have been adapted to accommodate our changing perception of software, a new more easily maintained structure for web sites is required.

The problems identified above can be ameliorated or solved altogether by the centralisation of web content storage and the introduction of some basic, well established software maintenance principles. In the remainder of this paper, maintenance problems associated with the WWW are analysed in greater depth in the following section. The application of traditional reverse engineering techniques to existing web sites is covered in a further section. The results achieved are discussed and evaluated in a final section, where future work is also considered.

2 MAINTENANCE PROBLEMS ASSOCIATED WITH THE WWW

Similarities can easily be drawn between large web sites and large-scale software systems [1,2]. Large sites often contain many thousands of lines of 'code' split into many 'modules' stored in many places with large amounts of data often important to the owner of the site.

In most traditional software systems, the code is segregated from the data by data abstraction, and many systems are data processors in which the program only interacts with the data when that data is provided to it. However, due to the design of HTML, in web documents, the data is marked up by tags to format both its structure and style of presentation. Thus, the data is an integral part of the 'code', and this complicates web maintenance. Others have noted that the blurred distinction between data and software in the HTML model presents developers with some interesting and problematic consequences [1,7].

Due to the ease with which web pages can be generated using a variety of tools (MS Frontpage, Netscape Composer, Macromedia Dreamweaver to name but a few), web sites can be created by people with little or no formal knowledge of software engineering and in very little time. For small scale personal web sites, it is not important that the site is error free and up-to-date; however, for larger corporate sites, this is often key to the businesses' on-line success.

Despite being young, the WWW has grown and is still growing rapidly with the result that modern sites are as large and complex as large-scale traditional software. In 1996, the maintainers of the Microsoft web site[1] estimated that they maintained over a million pages with a predictable impact on maintenance [8].

The rapid growth of the WWW combined with the necessity for businesses to quickly deploy the sites results in sites being produced with little or no design and no concern for the maintenance issues. This has resulted in a state of poorly maintained sites [9].

Specific Problems Identified

In research carried out into the state of web sites [10,8,9], four common problems has been identified; they are as follows:

- broken links,
- incorrect or out-of-date data,
- inconsistent information, and
- inconsistent style.

Surveys of web users [4] have shown that inconsistency and inaccuracy of data rank alongside with broken links as major obstacles in the take-up of the web. Inconsistent style and poor navigation have also been cited as major problems that discourage users from continuing to use a site [11]. Research into these areas is developing alongside Human Computer Interaction and Usability research to try and develop guidelines and strategies that will help businesses to utilise the web more effectively [5].

Broken links

This especially common problem is due to links being explicitly "hard-wired" into the HTML of a web page no longer pointing to the target page and its contents. A broken link is usually a consequence of one of the following:

- the linked site has moved or closed down;
- the linked page has been removed or renamed;
- the link has been incorrectly specified (i.e. it is mis-spelt or simply nonexistent);
- the page exists, but the its access permissions have been wrongly set;
- the page exists but the relevant information has changed or been removed making the linked content no longer relevant to the originating page with a link to it.

Ways have been developed to minimise the first three situations described above. These fall into two categories:

[1] http://www.microsoft.com

Proceedings of the Eighth Working Conference On Reverse Engineering (WCRE'01)
0-7695-1303-4/02 $17.00 © 2002 IEEE

dynamic link storage and link testing tools.

Dynamic storage usually consists of a database that stores the links centrally [6]. The HTML page containing the link is dynamically generated with the data for the database when requested. This enables the maintainer to easily update all occurrences of a link in a site when the link changes. It also simplifies the process of re-structuring a web site.

Link testing is carried out to find links that are no longer valid [6]. Typical systems parse the HTLM document and systematically test every link to ensure that a valid page exists at the end of the link. These tools can be used in conjunction with dynamic storage to provide more comprehensive aid to maintainers.

The final two situations are almost impossible to prevent unless there is co-operation between the maintainers of the pages at both ends of the link. Typically this will only be possible for links within a site.

Incorrect or Out-of-date Information

One of the major reasons for businesses setting up web sites is that they have large amounts of data that changes too frequently for them to publish or disseminate effectively through standard media (e.g. booklets, mailings, retail outlets) [10]. By creating a web site, they can have one central information point and concentrating on keeping it up-to-date and as accurate as possible. However, problems of out-of-date information are still common [9] and many sites give no indication of the currency of their information, nor do they indicate the last time of up-date.

Inconsistent Information

Inconsistent information is separate from the previous category because, although it will invariably include incorrect information, in the case of inconsistent information, some of it may be correct. In a typical web site, the same information may be stored in one or more different places. If this information changes and the maintainer doesn't up-date all occurrences of the information, the inconsistencies will be introduced into the site [9]. Consequently some of the information will be incorrect and misleading [4].

These problems of information incorrectness and inconsistency have been addressed by the creation of server side scripting systems [7] connected to databases that can be used to dynamically generate pages containing the most recent and up-to-date information obtained from the database. This will be discussed in greater detail in the section on problems using database storage with the WWW.

Inconsistent Style

This is a separate consideration from the design of the user interface in so much as it is not concerned with the actual choice of style used, but rather the consistent application of the styles used. Research into Human Computer Interaction has shown that systems should use a consistent and standard style throughout [4]. However, it is not uncommon to find different fonts, font sizes, colours, backgrounds and other style variations within the same web site across several pages or even in single pages of the site. Obviously this gives the site an unprofessional image as well as causing problems for users with navigation and readability.

The Internet communities, and the standards bodies in particular, have started to make great efforts to counter this problem. For instance, the consortium in charge of standardising the web, the W3C[2], has introduced the concept of Cascaded Style Sheets (CSS) which have been widely accepted by web site authors and browser developers alike.

CSS enable the author or author tool to abstract the style, and to a lesser extent the structure, away from the contents of the HTML file. Dave Raggett, one of the leading proponents of CSS and a major contributor to the W3C recommendation[3], has demonstrated the effective transformations that can be carried out by basic scripting and CSS [12]. By creating a CSS file for a collection of pages and making small changes to that single file, it is possible to dramatically alter the appearance of all the HTML pages in a consistent manner. This feature is of great importance to maintainers [14] as it minimises their maintenance efforts with respect to site style.

Root Causes of Web Site Maintenance Problems

While the above problems all reflect poor maintenance practices, they are symptomatic of poor design decisions and a general lack of Software Engineering principles being applied in web developments. In addition, several features of the existing WWW architecture compound the situation. Most notable among these are:

- forced duplication of files or data;
- the file structure of web sites; and
- the HTML format of combined code and data storage.

Duplication of Files or Data

If the same information is required in two or more pages, it is common for that information to be duplicated and placed on all the pages. HTML lacks an "include" directive common in many other languages. This problem is caused by the method of storage within HTML and forces the designer to choose between linking pages and copying the data. Likewise if a file, typically an image file, is needed in several places and it is awkward to reference it in a single location (see The Structure of Web Sites below), then

[2] http://www.w3c.org

[3] http://www.w3c.org/TR/REC-CSS2

Proceedings of the Eighth Working Conference On Reverse Engineering (WCRE'01)
0-7695-1303-4/02 $17.00 © 2002 IEEE

copies may be made and placed throughout the site. This causes the problems associated with the failure to successfully update all copies.

The File Structure of Web Sites
Web sites consist of files containing the site contents. The predominant file format is HTML, with some embedded content such as images and increasingly the inclusion of scripts both as part of the HTML document or as a means of generating the HTML page.

The files, which can be likened to modules in a conventional software system, are usually stored in directories to further "modularise" the site into comparmentalised blocks of similar data type or related content. When this approach is followed uniformly and sensibly across the site, it is very useful and effective in creating a site whose content is easy to manage [1]. However, the huge scale of some web sites and the inherent complexity of so many interlinked files makes the organised decentralisation of the pages a complex task which, if not carried out carefully and with planning, can result in illogical arrangements of files which quickly become too complicated to manage.

For instance, in a small site, it is sensible to store all images required on the site in an *images* directory and to use relative URLs to reference them. A relative URL is preferable to an absolute URL because pages can easily be moved around, even between web sites, without affecting the links as long as the file structures are preserved. However, as the site grows and its file structure becomes more differentiated, this can result in awkward use of relative file referencing and the loss of the advantages associated with relative URLs.

The HTML format of combined code and data storage
As noted earlier, HTML files are a combination of data and tags. The tags control both the style and layout of pages. As these contents are intermixed, a maintainer editing a page to alter content can inadvertently alter tags.

In summary, it is important to carefully design the file structure associated with a web site and to consider the implications of file contents on the maintainability of a site. This is an area of web development that has received little study until recently. In the following section, consideration is given to this area and the proposed solution of using a centralised data storage medium such as a database, as suggested in [13], is investigated.

Problems of using Database Storage with the WWW
One of the main uses for databases on the WWW is for the storage of company data such as customer or product details. This information may be stored in specialised databases designed and used solely for the company web site, or it may utilise a connection to existing company databases to produce the same result [7]. Obviously it is wasteful to maintain the same information in more than one database although for security reasons it may be sensible to make one database a partial or complete replica of another.

Many companies and organisations holding information in existing databases have sought to integrate these data stores with their web sites, rather than incur the penalties of having to maintain separate data sources. Dynamic page generating systems and associated scripting languages make such integration possible although considerable re-development of existing web pages may be necessary. In addition, co-ordination of the database maintenance with the web site maintenance is required.

Database systems can also be used to store more complex components of web sites or references to these. Examples of these are databases to store links [6, 14]. These enable maintainers to easily verify and update links throughout a site. However, in the case of older web sites, some reverse engineering of the site may be required to ensure that currently replicated elements within the web site are identified, abstracted, and, where appropriate, moved to a database or subsumed into a scripting language program.

3 REVERSE ENGINEERING OF WEB SITES

The databases used as discussed above are typically relational databases. Typically little or no modification of the existing database structures is required before integration with the web; however, complex scripts or programs may be required to extract, process and present the relevant information.

Many scripting languages have been developed to facilitate dynamic page creation. Several of these have had extensions built on to enable database access or have been designed with database integration in mind. An example of the latter is Microsoft's Active Server Pages (ASP); and an example of the former is the perl scripting language.

Databases employed in the fashion need not be closely integrated into the Web architecture as the scripting languages are well enough developed to make an almost seamless integration.

Given the investment of companies in their existing databases and current web pages, this research has addressed the possibility of reversing engineering existing web pages to identify replicated contents that can usefully be stored in databases where the databases will either already exist or be specifically created for this purpose. Through this research, the aim is to determine how to assist companies in developing a better basis for the future maintenance of their web sites and to overcome some the problems associated with web site maintenance discussed earlier.

The reverse engineering undertaken with respect to web sites has drawn on earlier work with the Reverse Engineering community dealing with code analysis and clone detection as a goal of the work is to detect and

Proceedings of the Eighth Working Conference On Reverse Engineering (WCRE'01)
0-7695-1303-4/02 $17.00 © 2002 IEEE

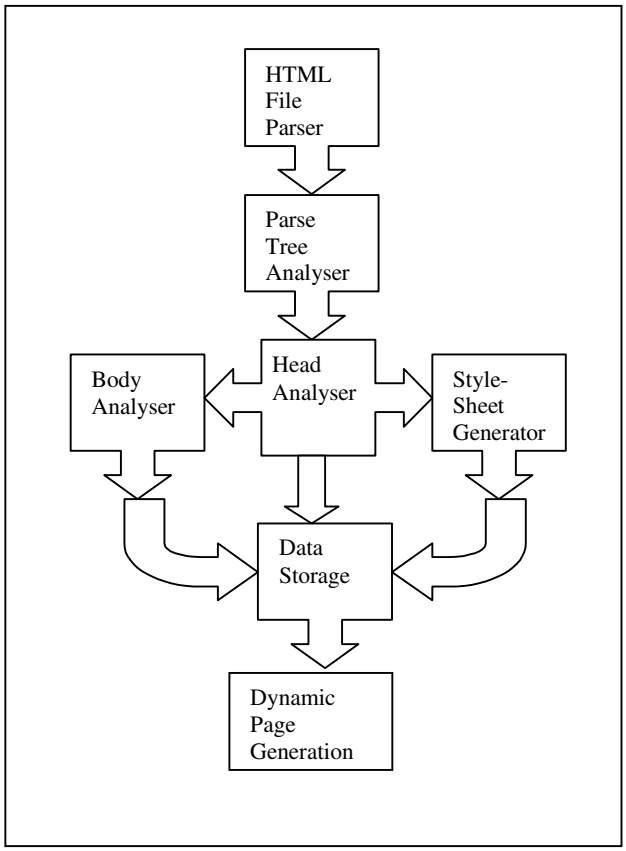
minimise duplication of web site content. In classical software engineering, a clone is any piece of code that has been copied and possibly altered. Clones are characterised by a measure of similarity ranging from 100% for replicas to 0% where no similarity can be found. However, as Baker suggests it is very hard to find two pieces of code that will not have some similarity, the sophisticated part is finding code that is a proper clone rather than simply being coincidentally similar [15].

For example, a designer might create several HTML pages on a web site, all with the same head section by copying the head from one page into all the others. This would create clones with 100% similarity. If subsequently, the titles of each page are altered, then the clones would not be identical although they would still have a high similarity.

Given that the alteration of web pages is not random and is likely to involve identifiable components, such as the page titles in the above example, the approach adopted here has been to parse and analyse the web pages adapting approaches for existing work in Reverse Engineering applied to software code. An approach employing metrics based on [16] is ruled out by the current lack of appropriate metrics defined for web site contents although this is certainly something to consider in the future.

Architecture of the System Developed

Below is an overview of the high-level architecture of the system developed.

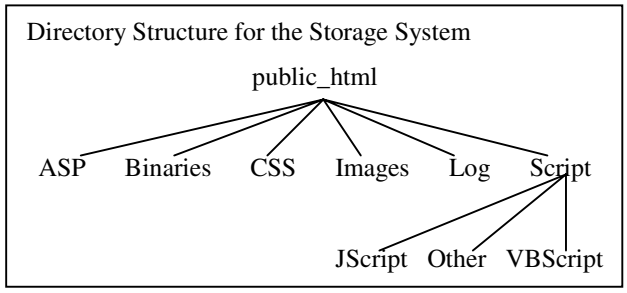


To achieve adequate coverage, it was decided that the parser within the system should handle pages that conformed to the HTML 4.0 specification with some additional features such as Active Server Pages (ASP), scripting languages (e.g. Javascript and VBScript) and Cascaded Style Sheets (CSS).

Although the W3C DOM Parser for HTML 4.0 was available, our system is based on an HTML 3.2 compliant parser from the JavaCC distribution. Unlike the W3C parser, this is not a validating parser, so it can accommodate invalid HTML. It also produces useful parse-tree output for further processing and thus, is a better basis for the replica detection. It is also easy to maintain and adapt due to its yacc-like design; extensions to HTML 3.2 have been easily achieved through altering the grammar supplied.

After parsing, the system stores a rationalised copy of the extracted HTML data in a central data storage with each unique element (e.g. Page titles separate from links) stored in a separate part of the repository. Content that cannot easily be stored (e.g. images) are stored in files within an organised directory structure. Rather than storing the details of how to reconstruct the pages, the original HTML page is modified to use scripts that retrieve the data from storage and regenerate pages as required. These modified files contain only the structure of the original HTML files, with all the content abstracted out into the data storage and the styles in CSS files. The files are stored in an organised directory structure on the server. It is these pages that will be accessed by the server, executed, and published when requested.

This approach was chosen because it could be implemented on top of existing web architecture. It also has the advantage of creating new web pages and an associated database that is both easy to maintain and limits the maintainer to altering data at its source, thereby ensuring that, because the pages are generated from source, they will always have the most up-to-date data content available. The database is supported by an organised directory structure that is used to store the external files. Several types of files are stored by the system including applets, images, scripts, and CSS files. The directory structure is created within a directory specified by the user and consists of the following directories listed in the figure below.



Proceedings of the Eighth Working Conference On Reverse Engineering (WCRE'01)
0-7695-1303-4/02 $17.00 © 2002 IEEE

Database Design
Having decided to use a relational database for the central data storage, the tables and fields were designed. The first consideration was given to what extra data should be stored in the database to facilitate easy maintenance in the future. From our earlier considerations, it was decided that the following should be added:

- the last date of update for every element as this enables the maintainers to track information that is out-of-date by a given date. Adding finer grained time stamps such as the time within a day was not considered necessary, as most information does not change so frequently to warrant this.
- an identifier for the person who carried out the update. This enables maintainers to confer with one another regarding changes that have been made and to ensure their work is co-ordinated in a consistent manner.

A record of changes can then be made in enough detail for maintainers to ascertain who changed what when.

Thus, the basic requirement for every table is that it should have fields allowing the following: unique identification of every entry; identification of the date of the last update; and identification of the maintainer who last updated it.

The detailed record of changes made is written to text files rather than the database as this is primarily for auditing purposes.

As the system requires the maintainers to supply both a username and password before changes can be made to the database, it is possible to identify each maintainer uniquely. The usernames are stored in a separate table with a unique ID number, that is referenced by the other tables to identify them.

The other tables are as follows: Titles, Styles, Scripts, Objects, Links, Images, Content, Bases, ASP. These are described in greater detail in the following section.

What is stored in where?
There are three broad types of data extracted by the system, each requiring a slightly different method of storage. These are as follows: textual content, large text files, and binary files.

The textual content of the site is the information that is presented to the user as text. This can be short strings, e.g. the title, or long paragraphs of text, more commonly found in body. This content changes frequently and is very important to the site owner. Character Large Object (CLOB) storage is used for textual content. On most database systems, each CLOB field can store up to 2Gb of text.

As scripts and CSS files contain large amounts of textual data. This data is imported into the HTML file by reference. Although CLOB fields could have been used, this would introduce the problem of how to recognise when the client needs the files and how to pass the content as if it was a separate file. This is avoided by storing all linked files in the directory structure and storing a reference to the file in the database.

Binary files, i.e. Image files and Java applets, like the large text files containing scripts and styles, are stored in the directory structure, with a reference in the database.

Content Layout and Processing
The tags surrounding the content define the layout of the HTML page as it will displayed in the client browser. The introduction of new tags and CSS into HTML4 has allowed some of the structural layout to be taken out; however, HTML tags still produce most of the structure.

Originally it was decided that the structure should be extracted during analysis and stored separately along with the style and contents; and that all these stored elements would be used to recreate the page. However, after studying a number of existing web page layouts, including sites with consistent structures, it was not possible to find enough similarities to be able to break the structure down satisfactorily. For this reason, the system keeps the structure in place as found in the existing web page and only extracts the style and content. To facilitate this, the analyser constructs an HTML file as it processes the parse-tree. The tags are analysed and then copied to the new file, with the content that has been removed from them replaced by an embedded script statement for retrieving the data from the database. When the analyser has finished, the new web page file is written to the directory structure at the top level.

To detect and remove duplication, the system processes each piece of text as follows:

1. all excess white space is removed (as this is ignored by browsers),
2. a code is generated for the text,
3. a comparison is made with codes for pieces of text already processed, and
4. a new element with its corresponding code is only stored if no match has been found.

Statistics
It was felt that maintainers might find it useful to see some details of the web page analysis results and subsequent processing. Therefore, the system generates details of what went on during the parsing, analysis and storage.

A log file, with a time and date stamp for easy identification, is created during every run. Logs of all files parsed are made along with a summary of the analysis. All error messages reported to the user during parsing are also

Proceedings of the Eighth Working Conference On Reverse Engineering (WCRE'01)
0-7695-1303-4/02 $17.00 © 2002 IEEE

logged.

After each parse, a breakdown is presented. It includes the following counts:

- count of embedded files identified and how many are not already recorded, broken down into file types, e.g. images and scripts;
- count of links identified and how many were not new;
- count of blocks of textual content found and how many of these were not new; and
- count of files parsed and number of failures and problems encountered.

Recreation of the Web Pages

After the system has been employed, the pages need to be re-generated from the new web pages and the database upon requests made for the new web pages by clients. These new web pages contain scripts that access the database. One basic function is used to extract the relevant data; it simply retrieves contents from a specified table and field. The returned string is appended to the end of the HTML stream being generated by the server as it interprets each script in the new web page.

There is an overhead for the script interpretation and the database accesses at the page load time. This differs in individual cases; and if the overhead is considered too great, it would be possible to recreate a new static web page after processing. However, this would require strict discipline to ensure that no changes were made to the static pages and changes were made through the new system with new versions of the pages being regenerated after any changes. Quite a sophisticated extension to the system would be required to determine the impact of any change on all the other pages in the site to ensure that changes were made uniformly across a site.

The advantage of using the dynamic approach is that changes made to the stored database associated with a set of web pages take effect immediately. Because all the data is stored within the database and no HTML pages are ever stored locally, the maintainers cannot make changes to the pages without changing the database thereby ensuring the entire set of pages is uniformly updated.

4 RESULTS OBTAINED

The system has been evaluated with a small sample of web sites:

1. Palatinate'99 - the web site of the student newspaper of the University of Durham. A snapshot of the site was taken in June 1999. The site consists of 121 HTML files in 26 directories with many scripts and linked files.
2. SEG'99 - The Computer Science Department at Durham have a web site of material associated with the second year Software Engineering group project which is part of the Software Engineering module. A copy of the site at June 1999 was studied. The site consists of 60 HTML files in 23 directories with scripts and linked files.
3. Palatinate'00 - This site replaced the Palatinate'99 site and is a re-designed site. The copy of the site studied was made in March 2000. The new site consists of 34 HTML files in 12 directories. It contains no scripts but does contain many linked files.
4. Personal Web Site 1 - This site was the homepage of a University of Durham student. Although this type of site would not normally be subject to formal maintenance, it is a good example of a well designed site with consistent features and use of scripts. The site is entirely HTML4 compliant and utilises CSS. It is well maintained and organised. The site consists of 27 HTML files in 5 directories.
5. Personal Web Site 2 - this is the old site of the same student. The copy studied has been replaced by Personal Web Site 1. It uses a combination of scripts, frames and images; but has been constructed with little regard to design principles and exhibits poor layout. The site consists of 16 HTML files in 2 directories. It contains several linked files.

Duplication Detected

The table below provides evidence of the level of duplication found and eliminated by the system.

Web site	Number of Pieces of Content Found	Average Number of Pieces of Content Stored
Palatinate'99	907	355
SEG'99	637	245
Palatinate'00	357	186
Personal Site 1	284	201
Personal Site 2	102	91

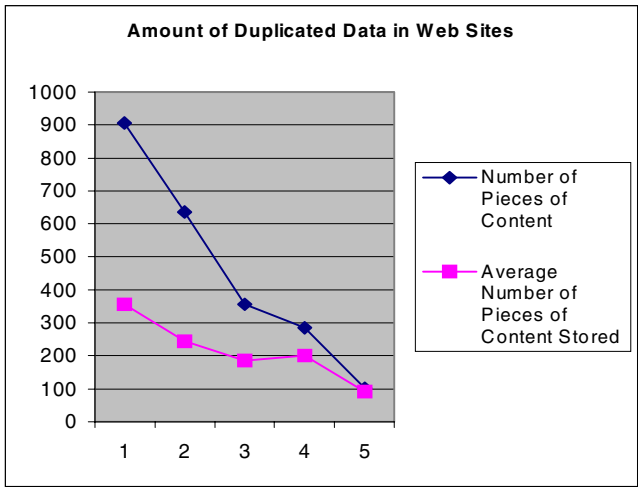
Note that while the parser consistently identified the same number of pieces of data for each site analysed, insignificant variations in the number of duplicated pieces were found depending on the version of the system used, so an average is given above.

The figure below shows these results graphically. It can be seen that there is a direct correlation between the number of pieces of content in the site and the amount of duplication detected. The more data there is the more duplication has been detected.

Conversely the smaller a site is, the amount of duplication

Proceedings of the Eighth Working Conference On Reverse Engineering (WCRE'01)
0-7695-1303-4/02 $17.00 © 2002 IEEE

is also smaller. This is shown by the narrowing gap between the two lines. For instance, on the larger Palatinate'99 site, only 355 pieces of data are stored from a possible 907. This means the system detected that almost two thirds of the data has been duplicated. However, on the smaller site, the proportion was found to be much lower. The second personal site has almost no duplicated data (only a tenth is duplicated) and the slightly larger and better designed personal site has only a third of its data duplicated.



This result is not surprising because the maintainers of small sites will be able to remember a higher proportion of the site than the maintainers of large sites, so they are more likely to realise when they are repeating the content.

Overall, the results show that the system does a good job of removing duplication within a web site, and thus with duplication minimised, the new site should be easier to maintain. Although there is an overhead, in generating the pages dynamically in the present system, it would be possible to generate static pages form the databases and use these provided that no maintenance was carried out on these directly as all the advantages of removing duplicates would be lost.

An extension to the system ensures timely up-date by monitoring changes made to the web site content databases and informing the maintainers every time that a data item over a specified age has been accessed. This feature provides a very useful service for the maintainers because it informs them if they have neglected some content for a long time. It also highlights contents that are being accessed most frequently; and, if necessary, they can focus their maintenance efforts on some preventative maintenance to ease the future maintenance.

5 CONCLUSIONS

This research has demonstrated that a significant reduction in the duplicated content of web sites can be achieved through application of the system described here. The system achieves this through a more rational re-structuring of the original web site pages and their contents. Identifying common styles and other contents requires extensive analysis of the existing web pages. However, the rationalisation of the site achieved should in theory allow maintainers to understand the contents of the web sites more straightforwardly and to separate concerns of style from other types of information content during subsequent maintenance of the site. Empirical evidence of such benefits can only be obtained in the longer term. At present the system developed has only been used in trials with a very small number of web sites. Integration of such a system with one of the more popular web page development systems would provide the necessary employment of the system by practitioners needed to form the basis for a more in-depth evaluation. We are considering this as a future development of the system.

The modest results achieved do clearly show that techniques from conventional Reserve Engineering can be applied to alleviate some of the maintenance problems found with existing web sites and the current storage systems employed for web sites. Certainly this work has shown that there is scope for further research on the reverse engineering of web sites.

REFERENCES

1. Warren, PJ, Boldyreff C, Munro M, The Evolution of Websites, International Workshop on Program Comprehension, IEEE Computer Society Press, 1999.
2. Brereton P, Budgen D, Hamiliton G, Hypertext: The next Maintenance Mountain, IEEE Computer, Vol. 31, No. 12, pp. 49-55, 1998.
3. Lehman MM, Belady L, Program Evolution: Processes of Software Change, Academic Press, London, pp. 247-274, 1985.
4. White MD, Abels EG, Hahn K, Identifying user-based criteria for Web pages, Internet Research, Vol. 7, No. 4, pp. 252-262, 1997.
5. White MD, Abels EG, Hahn K, User-based design process for Web sites, Internet Research, Vol. 8, No. 1, pp. 39-50, 1998.
6. Arnold SC, An Architecture for Maintaining Link Structure of a Website, Proceedings of WSE'99, 1st Annual Workshop on Web Site Evolution, pp. 9-11, 1999.
7. Antoniol G, Canfora G, et al, Web Sites: Files, Programs or Databases?, Proceedings of WSE'99, 1st Annual Workshop on Web Site Evolution, pp. 6-8, 1999.
8. Prevelakis V, Managing large WWW Sites, Internet

Proceedings of the Eighth Working Conference On Reverse Engineering (WCRE'01)
0-7695-1303-4/02 $17.00 © 2002 IEEE

Research: Electronic Networking Applications and Policy, Vol. 9, No. 1, pp. 41-48, 1999.

9. Warren, PJ, Boldyreff C, Munro M, Characterising Evolution in Web Sites, Proceedings of WSE'99, 1st Annual Workshop on Web Site Evolution, pp. 46-48, 1999.

10. Aspden P, Katz J, Motivations for and barriers to Internet usage, Internet Research: Electronic Networking Applications and Policy, Vol. 7, No. 3, pp. 170-188, 1997.

11. Nielsen, J, Designing Web Usability, new Riders Publishing, 2000.

12. Raggett D, Adding Style and Behaviour to Web Pages with a Dash of Spice, Computer Networks and ISDN Systems, Vol. 30, pp. 676-678, 1998.

13. van Ossenbruggen J, et al, Requirements for Multimedia Markup and Style Sheets on the World Wide Web, Computer Networks and ISDN Systems, Vol. 30, pp. 694-696, 1998.

14. Hartman JH, et al, Index-based Hyperlinks, Computer Networks and ISDN Systems, Vol. 29, pp. 1129-1135, 1997.

15. Baker, SB, On Finding Duplication and Near-Duplication in Large Software Systems, Proceedings of the Working Conference on Reverse Engineering, IEEE Computer Society Press, 1995.

16. Mayrand J, Leblanc C, Merlo EM, Experiment on the Automatic Detection of Function Clones in a Software System Using Metrics, Proceedings of the International Conference on Software Maintenance, IEEE Computer Society Press, pp. 244-253, 1996.

Proceedings of the Eighth Working Conference On Reverse Engineering (WCRE'01)
0-7695-1303-4/02 $17.00 © 2002 IEEE

Proceedings of the Eighth Working Conference On Reverse Engineering (WCRE'01)
0-7695-1303-4/02 $17.00 © 2002 IEEE